Embedded Systems Design: A Unified Hardware/Software Introduction

Chapter 7 Digital Camera Example

Outline

- Introduction to a simple digital camera
- Designer's perspective
- Requirements specification
- Design
  - Four implementations

Introduction

- Putting it all together
  - General-purpose processor
  - Single-purpose processor
    - Custom
    - Standard
  - Memory
  - Interfacing
- Knowledge applied to designing a simple digital camera
  - General-purpose vs. single-purpose processors
  - Partitioning of functionality among different processor types

Introduction to a simple digital camera

- Captures images
- Stores images in digital format
  - No film
  - Multiple images stored in camera
    - Number depends on amount of memory and bits used per image
- Downloads images to PC
- Only recently possible
  - Systems-on-a-chip
    - Multiple processors and memories on one IC
  - High-capacity flash memory
- Very simple description used for example
  - Many more features with real digital camera
    - Variable size images, image deletion, digital stretching, zooming in and out, etc.

Designer's perspective

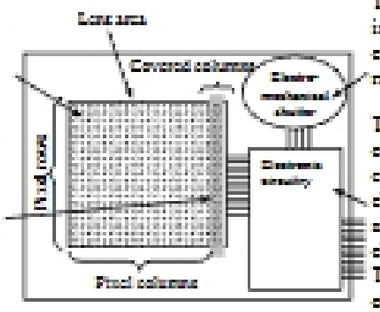
- Two key tasks
  - Processing images and storing in memory
    - When shutter pressed:
      - Image captured
      - Converted to digital form by charge-coupled device (CCD)
      - Compressed and archived in internal memory
  - Uploading images to PC
    - Digital camera attached to PC
    - Special software commands camera to transmit archived images serially

Charge-coupled device (CCD)

- Special sensor that captures an image
- Light-sensitive silicon solid-state device composed of many cells

When exposed to light, each cell becomes electrically charged. This charge can then be converted to a 8-bit value where 0 represents no exposure while 255 represents very intense exposure of that cell to light.

Some of the columns are covered with a black strip of paint. The light-intensity of these pixels is used for zero-bias adjustments of all the cells.

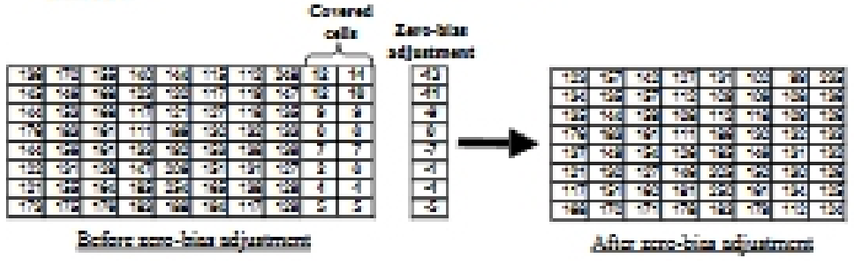


The electromechanical shutter is activated to expose the cells to light for a brief moment.

The electronic circuitry, when commanded, discharges the cells, activates the electromechanical shutter, and then reads the 8-bit charge value of each cell. These values can be clocked out of the CCD by external logic through a standard parallel bus interface.

Zero-bias error

- Manufacturing errors cause cells to measure slightly above or below actual light intensity
- Error typically same across columns, but different across rows
- Some of left most columns blocked by black paint to detect zero-bias error
  - Reading of other than 0 in blocked cells is zero-bias error
  - Each row is corrected by subtracting the average error found in blocked cells for that row



Compression

- Store more images
- Transmit image to PC in less time
- JPEG (Joint Photographic Experts Group)
  - Popular standard format for representing digital images in a compressed form
  - Provides for a number of different modes of operation
  - Mode used in this chapter provides high compression ratios using DCT (discrete cosine transform)
  - Image data divided into blocks of 8 x 8 pixels
  - 3 steps performed on each block
    - DCT
    - Quantization
    - Huffman encoding

DCT step

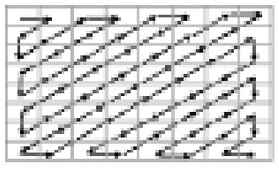
- Transforms original 8 x 8 block into a cosine-frequency domain
  - Upper-left corner values represent more of the essence of the image
  - Lower-right corner values represent finer details
    - Can reduce precision of these values and retain reasonable image quality
- FDCT (Forward DCT) formula
  - C(h) = if (h == 0) then 1/sqrt(2) else 1.0
    - Auxiliary function used in main function F(u,v)
  - $F(u,v) = \frac{1}{4} \times C(u) \times C(v) \sum_{x=0..7} \sum_{y=0..7} D_{xy} \times \cos(\pi(2u+1)u/16) \times \cos(\pi(2y+1)v/16)$
    - Gives encoded pixel at row u, column v
    - D_{xy} is original pixel value at row x, column y
- IDCT (Inverse DCT)
  - Reverses process to obtain original block (not needed for this design)

Quantization step

- Achieve high compression ratio by reducing image quality
  - Reduce bit precision of encoded data
    - Fewer bits needed for encoding
    - One way is to divide all values by a factor of 2
      - Simple right shifts can do this
  - Dequantization would reverse process for decompression

Huffman encoding step

- Serialize 8 x 8 block of pixels
  - Values are converted into single list using zigzag pattern



- Perform Huffman encoding
  - More frequently occurring pixels assigned short binary code
  - Longer binary codes left for less frequently occurring pixels
- Each pixel in serial list converted to Huffman encoded values
  - Much shorter list, thus compression

Huffman encoding example

- Pixel frequencies on left
  - Pixel value -1 occurs 15 times
  - Pixel value 14 occurs 1 time
- Build Huffman tree from bottom up
  - Create one leaf node for each pixel value and assign frequency as node's value
  - Create an internal node by joining any two nodes whose sum is a minimal value
    - This sum is internal nodes value
  - Repeat until complete binary tree
- Traverse tree from root to leaf to obtain binary code for leaf's pixel value
  - Append 0 for left traversal, 1 for right traversal
- Huffman encoding is reversible
  - No code is a prefix of another code